Stata Handout #1*

The purpose of this lab is twofold. First, I want to give you some basic familiarity with the Stata GUI (graphical user interface) and basic file and directory navigation. Second, I want to give you the opportunity to generate and visualize some descriptive statistics using a real social science dataset. I should note that I am writing these descriptions as a Mac user with a Mac version of Stata. Some things may be slightly different on a Windows machine, but we will cross that bridge when we get to it.

Section Ia: Stata GUI

When you first open Stata, you will see four "screens" of varying sizes. The one in the middle is where the results from your executed code will show up. You cannot edit it directly, but you can copy tables and images from it. The screen at the bottom is where you enter your "commands"—that is, the code (or syntax) you wish to execute. You type in your code, press enter, and the results are displayed in the screen above it. To the right of the command screen is another screen labeled "Variables." This lists the variables that you currently have loaded for potential analysis. Finally, just above the variables screen is a "Review" pane that logs the commands you have executed during that session.

Just above the middle screen is a series of icons. The two I want to draw to your attention are the "Data Editor" and "Data Browser" icons. If you click on the Data Editor one, you will see what looks a lot like a spreadsheet. This is where you input your data, and it is from this matrix that Stata performs your analyses. The Data Browser icon is the same thing, just without the editing privileges.

Another icon is the "Do-file Editor" icon. If you click on it, you will notice another tab open next to this document. These documents are called .do files, and they are where you store, organize, annotate, save, and share your Stata code. Though you can run any code you need from the command line on the main window, it is imperative that you make use of these .do files. The "Review" pane clears itself after every session, so you cannot use it as a reliable code manager. The .do file requires a little extra saving (since it is a separate file from the dataset with a distinct extension structure), but it is absolutely necessary for sound version control. You can—and should—annotate your .do file as you work, such as what I have done here. There are a number of ways to annotate, but I recommend using `//` and `///`. The former permits within-code notes, and the latter allows you to break code across lines.

Section Ib: File and Directory Navigation

You will want to have control over where files are stored. In the bottom left corner, you will see the directory in which you are working (for instance, my current directory is "/Users/mat4a/Google Drive/Summer2016_Bootcamp/"). Use the `-cd-` command to switch over to your N: drive:

```
cd N:/
```

The bottom left corner of your Stata window should now show that your current working directory is your N: drive.

---

* Thanks to Jon Schwarz and Brandon Sepulvado for some borrows language and code.

We can also use Stata to create a directory structure. Let's go ahead and make the directory structure that you will be using to complete your homework assignments and Stata projects using the `-mkdir-` function:

```
mkdir soc_30903
mkdir soc_30903/data
mkdir soc_30903/do_files
mkdir soc_30903/output
```

You should now see that, within your N: drive, you have a soc_30903 folder, and three sub-folders corresponding to the labels above. Now do the following:

```
cd N:/soc_30903/
```

From now on, your working directory should be the soc_30903 folder, and your will use the "data" sub-folder to store and retrieve your datasets, the "do_files" sub-folder to store your .do files (more on those later), and the "output" sub-folder to store your tables and graphs.

Now it's time to load up some data. Download the GSS 2014 file from Sakai and save it to the "data" sub-folder in your N: drive. Now type:

```
use data/GSS2014.dta, clear
```

You should have seen the "Variables" pane populate with variable names. Your data are now loaded. Note that "GSS2014.dta" portion of the code only works if your .dta file is actually called GSS2014.dta. Just match it to the file name if you saved it as something else.

Before we get to some analyses, however, it is important to note that it is generally not a good idea to work directly from the command line since the running record of what you execute from that space will disappear after you close your session. As such, we generally work from what are called ".do files," which are essentially just text documents where we write and execute our code. The nice thing about .do files is that we can save them separately from the .dta file and use them to re-run code at later dates. Let's set up a .do file using the `-doedit-` command:

```
doedit do_files/stata_handout_1
```

All of the code you write from now on will be written in a .do file such as the one that just opened in front of you. Simply highlight the code with your cursor, then click the "do" button in the tools ribbon. The results from your executed code will then display in the "Results" pane of the main Stata window.

Section IIa: Reporting and Visualizing Descriptive Statistics

Click on the Data Browder icon. You should see a populated spreadsheet. The top column consists of the variable names; the rows are individual respondents. Of course, it may be difficult to really understand what some of these variable names and cell values actually mean. A helpful command here is `-codebook-`, which you can use to display anything from response categories to the question asked. Let's use the command to get some information on the "zodiac" variable:

```
codebook zodiac
```

> **Thought Experiment #1:** What do we know about the "zodiac" variable? What type of variable is it?

Now let's get a little more information on the variable's descriptive statistics. We can get this information with the `-sum-` command (for "summary"), which lists the total number of non-missing values (i.e., the total number of respondents who answered the question), the mean, standard deviation, and range. We can add `, d` to the end of the command to list more detailed information. We can also use the `-tab-` command (for "tabulation") to get the frequency distribution and the associated percentages. Let's give both commands a try:

```
sum zodiac, d
tab zodiac
```

> **Thought Experiment 2:** What's the mean, median, and mode? Which of these measures of central tendency are appropriate given the type of variable? Why is this the case?

Let's try a couple more. Run the same commands with "wrldgovt" and "hivtest1":

```
codebook wrldgovt hivtest1
sum wrldgovt hivtest1, d
tab wrldgovt
```

> **Thought Experiment 3:** What type of variable is "wrldgovt"? What about "hivtest1"? How do you know? What are the appropriate measures of central tendency? What about dispersion? Why didn't I have you execute the `-tab-` command for "hivtest1"?

It is also helpful to visualize your data so that you can get a general idea of how they are distributed. The type of visualization you should use, of course, is largely dependent on the type of variable you have. Nominal and ordinal variables, for example, may be best conveyed through bar graphs. Let's make one with the `-histogram-` command:

```
histogram zodiac, discrete
```

Not particularly informative, is it? We can get some clarity with some extra code:

```
histogram zodiac, discrete freq xtitle("Zodiac Signs") ///
     xlabel(1 "Aries" 2 "Taurus" 3 "Gem" 4 "Canc" 5 "Leo" 6 ///
     "Virg" 7 "Lib" 8 "Scor" 9 "Sag" 10 "Cap" 11 "Aqua" 12 ///
     "Pisces") xscale(titlegap(2)) ytitle("Frequency") ///
     yscale(titlegap(2)) title("Zodiac Sign Frequency" ///
     "Distribution") graphregion(fcolor(white)) ///
     fcolor("maroon") barwidth(.9) saving(output/histo_zodiac)
```

Now let's visualize the "wrldfovt" variable in the same way:

```
histogram wrldgovt, discrete freq xtitle("International ///
```

```
     bodies should enforce environmental protection") ///
     xlabel(1 "Strongly A" 2 "A" 3 "Neither A nor D" 4 ///
     "D" 5 "Strongly D") xscale(titlegap(2)) ///
     ytitle("Frequency") yscale(titlegap(2)) title("") ///
     graphregion(fcolor(white)) fcolor("ltblue") ///
     lcolor("black") barwidth(.9) saving(output/histo_wrldgovt)
```

We can also use the -histogram- command without the "discrete" option to generate actual histograms (not bar graphs). This works best for continuous variables. Let's try it out on "cohort," which gives us the respondent's birth year. We'll include the "freq" option so that the y-axis is scaled by frequency and not density:

```
histogram cohort, freq fcolor("emerald") lcolor("black") ///
     xtitle("Birth Year") graphregion(fcolor(white)) ///
     saving(output/hist_cohort)
```

Sometimes we require something other than a bar graph or histogram. For instance, maybe the median is our best measure of central tendency and we want to get an idea for how the range and IQR are arranged. In that case we can create a box-and-whiskers plot. Let's use the "cutahead" variable, which is an ordinal variable asking respondents how often they have let a stranger go ahead of them in a line:

```
graph box cutahead, medtype(line) title("International ///
     Bodies and Environmental Protection") ///
     graphregion(fcolor(white)) saving(output/box_cutahead)
```

These are just a few of the possibilities!

> **Thought Experiment 4:** Take a look at the graphs and charts you just created (they should have been saved into your working directory). What do they tell you about the variables? Would you have chosen another visualization strategy? Why? What do they tell you about the variables' central tendencies and dispersion?

Section IIb: Go Forth and Explore

Now try running some descriptive statistics and visualizations on your own. Use the `-codebook-`, `-tab-`, and `-summarize-` commands as you see fit. Try to pick at least one nominal, ordinal, and continuous variable. Then go through the Thought Experiment exercises again with these new statistics, graphs, and charts in mind. Type your codes into a .do file rather than from the command line in the main window.

Also feel free to explore different commands and visualizations that you think may be appropriate. If you have questions while you do this, just explore Statalist (www.statalist.org) to see if your question has been answered. You can also type `-help-` if you know the appropriate command but not how to structure the code:

```
Help command
```

Or you can always just use Google.

Save your .dta and .do files when you are finished (you have to save them separately). You can do this directly from your .do file:

```
save "data/GSS2014.dta", replace
save "do_files/stata_handout_1.do",replace
```

Once you are done, execute the following to close you Stata session and exit the program.

```
clear
exit
```